UQE: A Query Engine for Unstructured Databases

Hanjun Dai[†♪], Bethany Yixin Wang[‡♪], Xingchen Wan[‡], Bo Dai[†¶], Sherry Yang[†],
Azade Nova[†], Pengcheng Yin[†], Phitchaya Mangpo Phothilimthana[†,*]
Charles Sutton[†], Dale Schuurmans[†§]
[†] Google DeepMind [‡] Google Cloud [§] University of Alberta [¶] Georgia Institute of Technology

Abstract

Analytics on structured data is a mature field with many successful methods. However, most real world data exists in unstructured form, such as images and conversations. We investigate the potential of Large Language Models (LLMs) to enable *unstructured* data analytics. In particular, we propose a new Universal Query Engine (UQE) that directly interrogates and draws insights from unstructured data collections. This engine accepts queries in a Universal Query Language (UQL), a dialect of SQL that provides full natural language flexibility in specifying conditions and operators. The new engine leverages the ability of LLMs to conduct analysis of unstructured data, while also allowing us to exploit advances in sampling and optimization techniques to achieve efficient and accurate query execution. In addition, we borrow techniques from classical compiler theory to better orchestrate the workflow between sampling methods and foundation model calls. We demonstrate the efficiency of UQE on data analytics across different modalities, including images, dialogs and reviews, across a range of useful query types, including conditional aggregation, semantic retrieval and abstraction aggregation.

1 Introduction

Data analysis [13] is essential for making well founded decisions and enabling businesses and society to function more effectively. Relational databases [12, 32] and the Structured Query Language (SQL) [7] have delivered huge successes in *structured data* management and analysis. Typically, such data is collected and organized in a pre-defined schema [14], where the data properties and relationships have been pre-specified, and downstream analysis is restricted to this schema.

In most real-world applications, however, data exists in unstructured formats, such as images, documents and audio recordings. Without preprocessing such data into structured forms, traditional SQL engines can only support limited queries. Preprocessing, including document entity retreival [45] and form understanding [42], also require training on downstream tasks given a predefined taxonomy. This naturally motivates the question we consider in this paper:

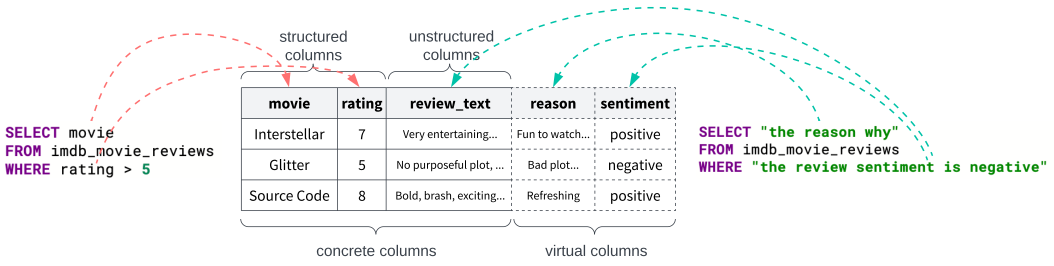
How can one perform unstructured data analysis in a **flexible** *and* **efficient** *way?*

In the literature, full-text search engines [20] support scalable regexp-matching search on unstructured data, but this becomes infeasible for more complex semantic reasoning queries. Retrieval-Augmented Generation (RAG) [39, 24, 18] allows question answering on a subset of related data, but is not directly applicable to generic analytical tasks with aggregation and semantic queries that spans over an entire large database. Recent advances in Large Language Models (LLMs) [4, 2] unlock the ability to perform flexible question answering, especially with recent long-context models [33]. However, setting aside the cost per query, data analytics can still be challenging for LLMs without fine-tuning [25] or few-shot demonstrations [9], even given structured tables.

[*] work done while Phitchaya Mangpo Phothilimthana was at Google DeepMind.

[♪] Correspondence to `hadai@google.com` and `wyixin@google.com`.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).



```
SELECT movie
FROM imdb_movie_reviews
WHERE rating > 5
```

movie	rating	review_text	reason	sentiment
Interstellar	7	Very entertaining...	Fun to watch...	positive
Glitter	5	No purposeful plot, ...	Bad plot...	negative
Source Code	8	Bold, brash, exciting...	Refreshing	positive

```
SELECT "the reason why"
FROM imdb_movie_reviews
WHERE "the review sentiment is negative"
```

Figure 1: Illustration of unstructured data analysis defined in Section 2.

Recently, a promising line of work considered marrying LLMs with programming frameworks [38], where logical or arithmetic operations are offloaded to program interpreters [17]. A most relevant example for analytics and table understanding tasks is Cheng et al. [11], which augments classical SQL semantics with LLMs as user-defined functions (UDF). While promising, the execution of such SQL programs that embed LLM calls still requires sweeping over the entire database, which is too costly for large collections of unstructured content. To overcome this barrier, we leverage the synergy between LLMs and programmatic execution to define an Unstructured Query Language (UQL) that augments SQL for flexible semantic queries, with a focus on improving scalability and efficiency.

A key observation is that the efficiency of classical SQL engines relies on (1) indexing structures that avoid the need to scan the entire database, and (2) a compilation system that determines the best execution order for operations. Based on these ideas, we propose the Unstructured Query Engine (UQE), which refines and extends this design principle to unstructured data analytics. To achieve similar effect to indexing, UQE casts the problem as learning to search or sample, seeking to avoid a full database scan with statistically sound methods. Additionally, a compilation system is developed that determines the best execution order and operator combination for different clauses in a UQL query, with the goal of minimizing LLM calls while preserving query semantics.

As part of this project, we have created four new benchmark datasets, with both text and image modalities, along with three common analytic tasks. Compared to baseline methods, such as long-context LLMs and embedding based retrieval, UQE achieves significant improvements in terms of the accuracy and cost reduction on these benchmarks.

2 Problem

Before defining the problem we are solving, we first establish the terminology and notation we will use throughout the paper. A concrete illustration of the following terms is given in Figure 1.

- **Table / database**: We define a table $\mathcal{T} = \{\mathcal{T}_i\}_{i=1}^N$ as an unordered set of $|\mathcal{T}| = N$ rows, where each row $\mathcal{T}_i = [\mathcal{T}_{i,1}, \mathcal{T}_{i,2}, \ldots, \mathcal{T}_{i,M}]$ is an array of M elements such that M is the total number of columns in the table. Each row can consist of elements $\mathcal{T}_{i,\cdot}$ of heterogeneous types (*e.g.*, datetime, float, enum) with different modalities (*e.g.*, text, image), while elements in each column $\mathcal{T}_{\cdot,j}$ must be of the same format and modality.
- **Structured data**: A column $\mathcal{T}_{\cdot,j}$ is structured w.r.t. a query if it can be accessed quantitatively, such as by algebraic operations over numeric data, comparison over string labels with predefined vocabulary (*e.g.*, categorical labels), datetime functions, *etc.*.
- **Unstructured data**: A column $\mathcal{T}_{\cdot,j}$ is unstructured if a query cannot access it using standard quantitative access. Typically, such a column does not belong to a predefined taxonomy. Examples include text (*e.g.*, dialogs), images, videos, and other forms of data that usually require semantic understanding and preprocessing before performing any algebraic operations.
- **Concrete column**: A column is concrete if it already exists in the table.
- **Virtual column**: A column is virtual if it does not already exist in the table, but a query is able to operate on it. Conceptually, one needs to derive (partial rows of) these columns by processing the data from concrete columns. In our work, we bypass this step by creating such columns lazily and selectively, which is the key to achieving efficiency and performance gains.

SQL engines can perform analytic queries on databases by manipulating *structured* data in *concrete* columns. The focus of this paper is to propose a new query engine that can perform analytics on databases with both *structured* and *unstructured* data, with queries that operate over both *concrete* and *virtual* columns. Standard analytics tasks that we seek to enable over unstructured data are:

- **Conditional aggregation**: perform aggregation operations on a sub-table filtered by a condition.
- **Semantic retrieval**: collect relevant rows specified by semantic filters.
- **Abstraction and aggregation**: group the rows based on abstractions and then performs aggregation.

Since optimizing queries on structured data within concrete columns is well-studied, we focus instead on techniques for handling queries on *unstructured* data over *virtual* columns. However, the UQE implementation also supports operations over structured data within concrete columns. In the following, unless stated otherwise, the term *unstructured* databases refer to databases containing *both* structured and unstructured data.

3 Unstructured query language

First, we need to formally define the query language, UQL, that talks to the unstructured databases. The idea of defining a natural query language for unstructured data is not completely new (*e.g.*, Cheng et al. [11], even though UQL has richer semantics), nor is the specific syntax or design of UQL the main focus of this paper. However, we need to define the scope of queries that the engine can handle, and breakdown the semantic meaning of each clause.

3.1 UQL semantics

We assume a basic familiarity of SQL, upon which UQL is based. UQL can be considered to be a dialect of SQL that has augmented functionalities for handling unstructured and virtual column queries. The SQL clauses that we support in UQL, along with necessary modifications to support unstructured semantic queries, are described as follows.

SELECT is a mapping function that maps the operand (usually a row or collection of rows in a grouped query) to a new row of elements. In traditional SQL, this mapping is usually a subset selection over concrete columns, or algebraic operators over those columns. UQL provides additional semantic mapping capability as:

```
SELECT "the attribute specified by natural language" AS attribute_name
```

For example, one can write `SELECT "the sentiment of the movie review"` over an unstructured movie review column, and retrieve "positive" or "negative" as a structured output.

FROM specifies the source of the table. In SQL one can additionally specify table joins, but we limit our attention to sourcing from a single table in this paper.

WHERE intrinsically specifies a binary classifier over rows, which is used to retrieve a subset of the database. In addition to comparator operators on structured columns, we also allow semantic specifications in the form of:

```
WHERE "the row satisfies some natural language specifications"
```

The predicates in `WHERE` are organized in disjunctive normal form (DNF) with `AND` and `OR` syntax, so a user can arbitrarily express predicates over concrete and virtual columns.

GROUP BY partitions the table into groups, where rows within each group share the same attributes over the keys being grouped by. UQL allows partitioning over virtual columns via natural language:

```
GROUP BY "the abstraction criteria specified in natural language"
```

Similar to `WHERE`, one can `GROUP BY` over both concrete and virtual columns by concatenating multiple criteria, with the resulting partition corresponding to grouping by a tuple of these keys.

We also reuse other clauses from SQL including: **ORDER BY**, which simply inherits the SQL semantics to rank the resulting rows according to a specific *concrete* column. In most analytics tasks, sorting is applied over structured columns with well defined ordering comparators. **LIMIT** is applied during processing in the form of `LIMIT num_rows`, which limits the number of output rows.

Assumptions: We rely on the ability of an LLM to perform *intra-row* semantic understanding and analysis tasks. For example, we assume that LLMs are able to correctly judge the specification in `WHERE` for a *single* row. Similarly, LLMs should be able to extract the information specified by `SELECT` or `GROUP BY` for a *single* row. We build programmatic functionality on top of this fundamental ability of LLMs to handle analytics for large databases.

```
SELECT reason, COUNT(*) as count
FROM movie_reviews
WHERE movie_year < 2020
GROUP BY "the reason why the
          review is positive"
          AS reason
```

```
SELECT agent_name, "reason to cancel"
FROM airline_customer_service_log
WHERE "the customer asked to cancel
       the flight"
ORDER BY ticket_price
LIMIT 100
```

Figure 2: Aggregation (left) v.s. Non-aggregation (right) queries written in UQL.

3.2 UQL queries

A UQL query is a composition of clauses that can be categorized as an aggregation or a non-aggregation, as illustrated in Figure 2. **Aggregation** queries perform a summary on groups of aggregated rows, such as `COUNT` the number of rows, or summarize a common attribute in a group of rows (as defined in `GROUP BY` above). **Non-aggregation** queries perform operations on individual rows, which usually means the rows can be processed in parallel. A UQL query will only belong to one of the above two types and we do not consider nested queries for now. The query type determines how UQE will optimize and execute the query.

4 Unstructured Query Engine

One straightforward way to run UQL is to use an interpreter that executes queries imperatively. Cheng et al. [11] implements an engine in this form, which is able to handle tables of relatively small size. By analogy, this is similar to executing an SQL program using a linear database scan. While it is valid, the latency and cost are prohibitive and generally prevent scaling to real world scenarios.

There are (at least) two key techniques in SQL databases for making query execution efficient:

- **Indexing**, which organizes concrete columns via data structures for fast search with sublinear cost.
- **Compilation**, which considers alternative query plans and executes the most efficient one.

UQL queries over virtual columns pose challenges in both indexing and compilation. In this section, we present effective approaches to indexing (Section 4.1) and compilation (Section 4.2) for unstructured databases, along with the implementation details of low-level primitives (Section 4.2.2).

4.1 Indexing

Executing traditional SQL queries over indexed columns can be made efficient by avoiding an entire database scan to find the relevant rows to process. However, for UQL queries over virtual columns, it is hard to predict or predefine an index that can enable efficient searching, since these columns are not concrete and defined via arbitrary natural language specifications.

Our first key contribution is to introduce a proxy for "indexing" that allows one to leverage the intrinsic semantic content of a virtual column to efficiently execute queries without scanning the entire database. The main idea is to use statistically sound sampling techniques to approximately retrieve relevant rows for processing. Based on the two types of queries defined in Section 3.2, we develop corresponding "indexing" counterparts.

4.1.1 Unbiased estimation for aggregation queries

We use a simple query to illustrate the idea of unbiased estimation to obtain a query result without scanning over an entire virtual column.

```
SELECT COUNT(*) as count FROM movie_reviews WHERE "the review is positive"
```

Given a row $\mathcal{T}_i$ (a natural language movie review), it is relatively easy for an LLM to tell whether it satisfies the `WHERE` condition. If we use $f : (\mathcal{T}_i, \texttt{cond}) \mapsto \{0, 1\}$ to represent the LLM's classification of whether row $\mathcal{T}_i$ satisfies the conditions specified in `cond`, then the goal is to estimate the quantity

$$\sum_{i=1}^{N} f(\mathcal{T}_i, \texttt{cond}) = N \sum_{i=1}^{N} \frac{1}{N} f(\mathcal{T}_i, \texttt{cond}) = N \mathbb{E}_{i \in \{1 \dots N\}} \left[f(\mathcal{T}_i, \texttt{cond}) \right] \tag{1}$$

There are many approaches that can be used to estimate the finite sum in the above equation, with different tradeoffs between bias and variance. One unbiased but potentially high variance estimator is to simply use Monte Carlo samples from a uniform distribution over $1 \dots N$. A typical technique for reducing variance is to use importance sampling with a proposal p, according to

$$\mathbb{E}_{i \in \{1 \dots N\}} \left[f(\mathcal{T}_i, \texttt{cond}) \right] = \sum_{i=1}^{N} \frac{p_i}{N p_i} f(\mathcal{T}_i, \texttt{cond}) = \mathbb{E}_{i \sim p} \left[\frac{1}{N p_i} f(\mathcal{T}_i, \texttt{cond}) \right] \tag{2}$$

A theoretically optimal proposal p is given as follows:

Algorithm 1 Stratified sampling for unbiased aggregation

```
• Embed each row $\mathcal{T}_i$ as $\vec{e}_i$, using a multi-modal embedding over the unstructured columns of $\mathcal{T}_i$.
• Cluster the embeddings $\{\vec{e}_i\}_{i=1}^N$ into $K$ disjoint groups $\{C_k\}_{k=1}^K, C_k \subseteq \{1 \dots N\}$, where $k$ can be
  a predefined constant or automatically selected [48]. Each group has size $|C_k|$ and $\sum_{k=1}^K |C_k| = N$.
  We use $c : \{1 \dots N\} \mapsto \{1 \dots K\}$ to denote the cluster index of each row.
• Perform stratified sampling over these groups and obtain samples $S \subseteq \{1 \dots N\}$.
Then we can obtain the following estimator for the expectation:
  $\mathbb{E}_{i \in \{1 \dots N\}} [f(\mathcal{T}_i, \texttt{cond})] \simeq |S| \sum_{i \in S} \frac{w_i}{\sum_{j \in S} w_j} f(\mathcal{T}_i, \texttt{cond})$, where $w_i = \frac{|C_{c(i)}|}{\sum_{j \in S} \mathbb{I}[c(j) = c(i)]}$     (3)
```

Algorithm 2 Online active learning for non-aggregation retrieval

```
1. Embed each row $\mathcal{T}_i$ as $\vec{e}_i$, using multi-modal embedding over the unstructured columns of $\mathcal{T}_i$.
2. Maintain $\hat{g}(i)$ that approximates $f(\mathcal{T}_i, \texttt{cond})$. Initialize $\hat{g}(i) \propto U(0, 1)$ uniformly.
3. Maintain the collection of sampled rows $S = \emptyset$ at step $t = 0$.
4. At step $t$, obtain a batch $S_t$ of samples where $S_t = \arg\max_{S_t \subseteq \{1 \dots N\} \setminus S} \sum_{i \in S_t} \hat{g}(i) + \epsilon_{t,i}$;
   observe $f(\mathcal{T}_i, \texttt{cond})$ for each sample $i \in S_t$; Update $S \leftarrow S \cup S_t$.
5. Fit $\hat{g}$ with samples and corresponding observations from $S$. Go to step 4 if $|S| < B$ or return the
   positive samples found in $S$.
```

Proposition 1 *The optimal proposal distribution* p *that minimizes the variance of estimation in Eq 2 is* $p_i \propto f(\mathcal{T}_i, \texttt{cond})$*, which achieves zero variance.*

Prop 1 indicates that an ideal proposal should sample rows that have positive f with equal probability, while sampling negative rows with zero probability. However, given that f is the response of an LLM it is expensive to execute over all rows, forcing us to consider efficient approximations.

Stratified sampling leverages the ability to partition a population into homogeneous subpopulations. As shown in Prop 1, a good proposal p_i should predict whether a row $\mathcal{T}_i$ satisfies the target property cond. To trade-off between cost and variance reduction, we propose Algorithm 1. In Eq 3 we normalize the importance weights w_i to further reduce the estimation variance.

Extension The above estimator can be used for other aggregation operations such as SUM and AVERAGE, including GROUP BY, and allowing concrete columns as operands as well. However, some aggregations such as MAX does not admit such an estimator. UQE in this case can only provide estimates with greater effort. We discuss limitations in Section 7.

4.1.2 Online learning for non-aggregation queries

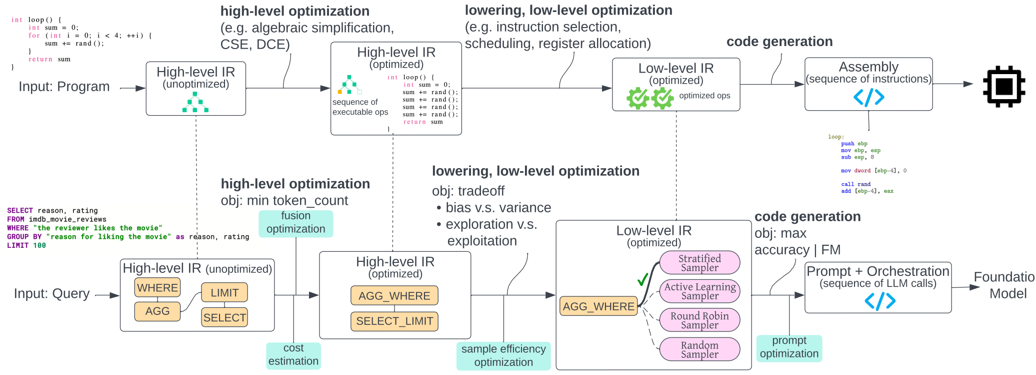
A non-aggregation query can be viewed as a search problem where we want to find a relevant subset of rows to process. As before, we begin with a concrete example:

```
SELECT dialog_ID FROM dialogs WHERE "the customer is unhappy with the agent's manner"
```

In practice, we aim to identify as many rows as possible that satisfy the given condition while adhering to budget constraints (e.g., the total number of tokens allowed to expense). This can be formulated as an online learning problem: given the token budget (or approximately, the number of LLM calls over individual rows, denoted as B), we seek to balance exploration (to better understand the semantic landscape of all rows) with exploitation (to maximize recall). Drawing inspiration from Bayesian optimization, which employs a surrogate model learned on-the-fly to inform sequential decision-making [36, 19, 22], we use a cheap proxy $\hat{g}$ as a *surrogate* for $f(\mathcal{T}_i, \texttt{cond})$. At each step t, we re-train $\hat{g}$ with the observed data and select its maximizer to query in the next step. See Algorithm 2. Here we use random noise $\epsilon_{t,i}$ to allow some degree of exploration that decays with t. Compared to the typical max inner-product search method prevalent in RAG systems, we rely on the online learning to adjust the beliefs, instead of solely relying on predefined embedding similarities.

4.2 Compilation

Classically, the goal of a compiler is to translate a high level program to low-level machine code, maintaining *exact* execution results with improved execution speed. Such a lowering process is usually accompanied by optimizations *e.g.*, fusing, selecting the optimal instructions, and kernel optimizations. Our goal is similar: we would like to compile high-level UQL into low-level machine code, with the distinction that the "machine" is an LLM, and the "low-level code" is the orchestration of prompt calls to the LLM. Given that the primary bottleneck is the LLM API calls, we attempt to maintain the execution semantics while minimizing the cost of LLM calls, as in Figure 3.



```
int loop() {
    int sum = 0;
    for (int i = 0; i < 4; ++i) {
        sum += rand();
    }
    return sum
}
```

```
int loop() {
    int sum = 0;
    sum += rand();
    sum += rand();
    sum += rand();
    sum += rand();
    return sum
}
```

```
loop:
    push ebp
    mov ebp, esp
    sub esp, 8

    mov dword [ebp-4], 0

    call rand
    add [ebp-4], eax
```

```
SELECT reason, rating
FROM imdb_movie_reviews
WHERE "the reviewer likes the movie"
GROUP BY "reason for liking the movie" as reason, rating
LIMIT 100
```

Figure 3: UQL compiler, in analogy to a typical C++ program compiler.

4.2.1 Planning

Lowering a query into sequences of concrete execution units is a planning problem: The action space includes the order of clause execution, as well as ways to fuse clauses to execute together. The objective is to minimize the (estimated) LLM cost. Figuring out the best decomposition and combination is usually an NP-hard problem. Fortunately, the number of clauses is very limited for a single query, so we can enumerate possible combinations of ordering and fusions with little overhead.

The outcome of planning is a specification of a sequence of kernel executions. The input and output of each kernel can be one of the following:

- **Concrete table**: a standard table with only concrete columns.
- **Stochastic table**: the outcome of unbiased sampling of a table. Importance weights will be attached to each row of the table, and the operation (*e.g.*, `SUM`, `AVG`) on this table takes weights into account.

In the following 3 sections we will explain the building blocks of the compiler, including the kernel implementation, the cost estimation and final instantiation in detail.

4.2.2 Kernel implementation

Each kernel is an standalone execution unit that reads and produces a (stochastic) table.

SELECT on structured columns is straightforward. When operating on unstructured columns, we prompt the LLM to extract semantic attributes from the input data. If several extractions share the same source column, we can also group these together into a single prompt to reduce cost.

WHERE takes a logical formula in disjunctive normal form, such that each conjunction can contain predicates over both unstructured and structured columns. One optimization we make in this case is to perform evaluations over the structured columns first, then simplify (*e.g.*, remove a conjunction if any of the structured column evaluates to false) the logical formula. Any remaining predicates over unstructured columns are then executed on the table filtered by predicates over structured columns.

GROUP BY first gathers a representative subset of rows from the table, then calls an LLM to extract a taxonomy (*i.e.*, the description of each cluster) for a cluster abstraction. Then the taxonomy is used to classify rows sampled according to the methods defined in Section 4.1. Finally, each row is classified into one of the clusters with the corresponding cluster description in the taxonomy.

Other standard kernels like `ORDER BY` are implemented as-is since they are efficient to execute.

Kernel fusion: Certain clauses can be fused together to achieve significant efficiency gains.

- `WHERE + LIMIT` can be terminated earlier for non-aggregation queries, once the number of rows specified by `LIMIT` are retrieved. This is particularly useful for rare event finding.
- `SELECT + GROUPBY` when executed together, the semantic attribute extraction of `SELECT` and taxonomy classification in `GROUPBY` can be done in the same LLM call to save cost.
- `GROUPBY + WHERE` can share the same sampling proposal for the aggregation queries.

When and how to fuse clauses relies on the planning technique introduced in Section 4.2.1.

4.2.3 Cost estimation for each kernel

We only consider the cost of calling the LLM, as this dominates the overall cost per query. Assuming the length of each row in the unstructured data is more or less uniform across rows, then the cost is proportional to the number of rows that fed to the LLM, which we use as the surrogate for estimation.

- SELECT maps each row, hence the cost is $|\mathcal{T}|$ for the table $\mathcal{T}$ fed to SELECT.
- GROUP BY consists of two steps, where taxonomy construction consumes a subset of the input table $\mathcal{T}$, and classification runs $|\mathcal{T}|$ LLM calls in parallel.
- WHERE depends on the proposal p. In practice we set a budget B and try to minimize the variance of unbiased estimator or maximize the recall in online learning, as explained in Section 4.1.
- Whenever clauses are fused together, each implementation is responsible for providing a reasonable cost estimate. For example when SELECT and GROUP BY are fused, the estimated cost is the same as GROUP BY alone, as the classification stage of GROUP BY shares the input tokens with SELECT.

4.2.4 Instantiation of kernels

The last step of compilation is to generate the machine specific code (*e.g.*, x86 assembly code) from the intermediate representations (IR). For UQE, this is the process of generating the LLM-specific prompts. For example, when GPT is deployed as the "machine", a system prompt like "You are a helpful assistant" will be added to the queries. This step also sets the correct context (*e.g.*, the correct structured/unstructured column to associate to, the description of the databases) for the LLM. When such information is not available, one can also leverage the LLM to provide a good suggestion.

5 Related work

While the unstructured data analytics engine is relatively new, there are several related works in the context of unstructured data query and analysis. Approaches like pattern or regexp matching [20] is scalable but not feasible for complex semantic reasoning. RAG [24, 18] based approaches rely on the retrieval quality and is not directly suitable for aggregation queries over entire database. LLMs [4, 2, 33] depict the ability of table analytics [15] to some extent [9, 25], but are still not reliable for large unstructured database analytics yet.

Our work is closely related to neural symboilic approaches for unstructured data analytics. Early attempts in this line aim to design specialized neural architectures with inductive biases (e.g., attention) to capture a particular form of operation (e.g., filtering a list of objects based on a natural language predicate by their attention scores) [43, 29, 3]. Those differentiable neural "operators" can then be chained together to model more compositional queries, and trained end-to-end using gradient descent. Another direction, in line with our work, is to augment symbolic programs with learnable operators parameterized by neural networks [10]. Those programs are often modeled as discrete latent variables, which can be hard to optimize. In contrast, UQE leverages predictions from LLMs as supervision to train an efficient proxy query model in an online fashion. Similar to UQE, some recent work [11, 38] also adopts LLMs as fuzzy query operators. However, the generated programs treat LLMs as an UDF in a SQL program, which can be very expensive to execute on large databases. Our UQE implements similar but augmented semantics with the focus on the cost efficiency and scalability. Liu et al. [26] optimizes a similar query engine from the system perspective like cache optimization and deduplication, while our work mainly considers algorithmic improvements and is considered as an approximate query engine [28]. These system and algorithm optimizations are actually orthogonal and can be beneficial to jointly consider both for future works.

In a distantly related topic, text2SQL [47, 46, 21, 35] also leverages models talking to databases, but is mainly for semantic parsing purpose. While it also leverages the advances in LLMs [44, 16, 31, 37], the execution is still on pure SQL and thus is not suitable for unstructured databases. There are also works on leveraging formal query languages to better query LLMs [34, 6], with the focus on controllability of the LLM itself rather than performing analytics on external unstructured data.

6 Experiments

We benchmark the accuracy and incurred cost of UQE on multimodal unstructured data analytics tasks, with the goal to show and understand when and why UQE can improve accuracy while keeping the cost low. Since the unstructured database analytics is a relatively new task, we construct and compare against several baseline approaches, on a set of tasks created from existing datasets.

Table 1: Conditional aggregation results on benchmark datasets. We report the relative error and the average cost per query. *gpt-4o is 50% cheaper than gpt-4-turbo so we double its budget of tokens; † we use claude-3-haiku as the backend LLM.

Benchmarks	Conditions	Methods			
		lc-gpt-4-turbo	lc-claude-3-opus	UDF†	UQE †
IMDB	`sentiment_positive`	49.02% ± 21.23%	56.05% ± 14.69%	13.67% ± 6.24%	**5.75% ± 3.43%**
Average cost per query		$0.37	$0.61	$0.01	$0.01
ABCD	`account_access`	69.25% ± 32.82%	27.28% ± 17.05%	18.99% ± 9.85%	**11.75% ± 9.78%**
	`single_item_query`	78.42% ± 9.36%	23.39% ± 16.14%	26.95% ± 22.16%	**12.32% ± 10.53%**
Average cost per query		$0.38	$0.63	$0.01	$0.01
AirDialog	`book`	47.58% ± 15.24%	54.40% ± 13.37%	10.15% ± 7.64%	**4.98% ± 2.26%**
	`no_flight`	47.92% ± 21.62%	53.88% ± 15.77%	21.08% ± 16.78%	**8.78% ± 8.12%**
	`no_reservation`	50.54% ± 21.86%	57.49% ± 13.08%	21.19% ± 12.10%	**7.23% ± 5.40%**
Average cost per query		$0.21	$0.36	$0.01	$0.01
		lc-gpt-4o	lc-gpt-4-turbo	UDF-gpt-4o	UQE-gpt-4o
Clevr	`obj_count < 4`	22.46% ± 19.35%	48.15% ± 41.52%	31.04% ± 25.15%	**9.55 ± 8.55%**
	`# spheres > 3`	35.72% ± 14.95%	91.09% ± 20.08%	19.35% ± 13.81%	**15.14% ± 10.71%**
Average cost per query		$0.33*	$0.33	$0.20	$0.20

Table 2: Semantic retrieval results on several benchmark dataset. We report the F1 score of the retrieved rows and the average cost per query. We run 8 independent queries and report the average F1 and its standard deviation. The result of MIPS is deterministic, so no standard deviation is reported.

Benchmarks	Conditions	Methods				
		lc-gpt-4-turbo	lc-claude-3-opus	UDF	MIPS	UQE
IMDB	`sentiment_positive`	0.397 ± 0.041	0.556 ± 0.066	0.505 ± 0.030	0.875	**0.978 ± 0.003**
Average cost per query		$0.38	$0.63	$0.02	≃ $0	$0.02
ABCD	`account_access`	0.045 ± 0.033	0.080 ± 0.023	0.076 ± 0.017	**0.961**	0.940 ± 0.019
	`single_item_query`	0.023 ± 0.021	0.082 ± 0.030	0.065 ± 0.017	0.266	**0.935 ± 0.006**
Average cost per query		$ 0.76	$1.23	$0.03	≃ $0	$0.03
AirDialog	`book`	0.327 ± 0.0667	0.585 ± 0.025	0.342 ± 0.031	0.930	**0.979 ± 0.010**
	`no_flight`	0.066 ± 0.037	0.228 ± 0.068	0.144 ± 0.034	0.867	**0.928 ± 0.018**
	`no_reservation`	0.156 ± 0.075	0.297 ± 0.043	0.145 ± 0.042	0.965	**0.969 ± 0.004**
	`cancel`	0.006 ± 0.009	0.013 ± 0.008	0.013 ± 0.009	0.066	**0.741 ± 0.205**
Average cost per query		$0.43	$ 0.74	$0.01	≃ $0	$0.01
Clevr	`obj_count < 4`	0.058 ± 0.026	0.066 ± 0.023	0.093 ± 0.031	0.023	**0.850 ± 0.025**
	`# spheres > 3`	0.037 ± 0.027	0.099 ± 0.023	0.089 ± 0.017	0.145	**0.633 ± 0.177**
Average cost per query		$0.38	$0.21	$0.08	≃ $0	$0.08

Baselines: We design the following baselines for comparison

- **lc-LLM** denotes the long-context LLMs that can directly take a subset of database and a natural language question as input, and produce the desired analysis. We mainly evaluate against several model families, including GPT-4 [2] and Claude-3 [4]. Of course, when evaluating the lc-LLM based approaches, we use the natural language instead of UQL as the prompt.
- **RAG-based** can be applied to some non-aggregation queries, such as semantic retrieval. For the retrieval part we use max inner-product search (MIPS) on top of the same embeddings that are used by UQE, for a controlled experiment.
- **UDF** simply treats the LLM calls as User-defined function of an SQL engine, with the same budget as UQE by default. This approach will not have the advanced sampling / search algorithm as used in UQE, which also serves as an ablation for the effectiveness of our UQE.

Datasets: We evaluate different approaches on common analytical tasks in three widely used application domains. We use the datasets that were previously created for discriminative tasks, as these datasets contain both the unstructured columns and the structured ones (the labels in the corresponding dataset). We then hide these structured label columns and perform analytical tasks on the unstructured columns, where these hidden structured columns will be used to compute the ground-truth. The text based tasks include IMDB [27] movie reviews, customer service dialogs including Action-Based Conversations Dataset (ABCD [8]) and AirDialog [41], and image based Clevr [23] dataset. Please refer to Appendix C.1 for more information.

Setup: We use voyage-2 [1] to embed the text-based unstructured columns, and Vertex [40] for multimodal embeddings. For budget constraint queries, we allow different approachces to access at most 128 rows in the database by default.

Table 3: Conditional abstraction and aggregation.

Benchmarks	Metrics	Methods		
		lc-gpt-4-turbo	lc-claude-3-opus	UQE-claude-3-haiku
AirDialog	`EMD`↓	0.143 ± 0.034	0.121 ± 0.014	**0.111 ± 0.019**
	`cost`	$0.21	$0.37	**$0.04**
ABCD	`account_access-EMD`↓	0.154 ± 0.031	0.113 ± 0.010	**0.110 ± 0.016**
	`single_item_query-EMD`↓	0.031 ± 0.034	0.011 ± 0.006	**0.005 ± 0.002**
	`cost`	$0.34	$ 0.56	$ 0.07

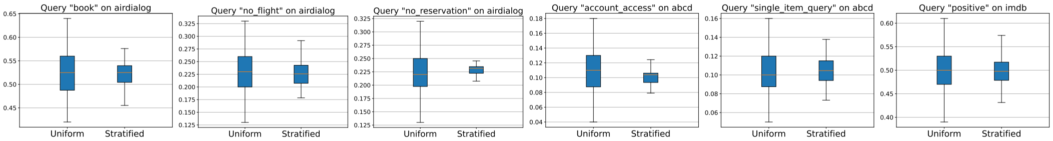


Figure 4: Variance of different sampling approaches for aggregation queries over 3 text datasets.

6.1 Main results

We run queries on different datasets by instantiating the template shown in each of the sections below. The exact natural language and UQL queries can be found in Appendix D and more information including statistics of conditions we used for query and hyperparameters (for UQE we simply use the default hyperparameters for sampling and online learning) can be found in Appendix C.

6.1.1 Conditional aggregation

This task provides aggregated statistics over databases with specified conditions, with the template as:

```
SELECT COUNT(*) FROM {table} WHERE {"satisfies natural language specified condition"}
```

We report the relative estimation error (*i.e.*, $|\texttt{predict}-\texttt{true\_count}|/\texttt{true\_count}$) and its standard deviation in Table 1. For lc-LLM baselines we estimate the count based on groups of unbiased data samples that fed into the prompt.

For text based aggregation we use claude-3-haiku as the backbone model, where UQE deploys 10× reduction in relative errors while reducing the cost by a factor of 20× or more. For the image dataset, since only limited set of LLMs are capable right now, we use gpt-4o as the backbone, and compare with lc-LLM baselines. Thanks to the improved sampling method in UQE, the same gpt-4o consistently achieves improved performance out-of-the-box. To verify this, we feed one image at a time to gpt-4o and manually aggregate the count, the estimation error would be 17.10% ± 13.95 and 19.35% ± 13.81% for the two queries of Clevr, which is twice higher than UQE in the worst case.

6.1.2 Semantic retrieval

This task filters rows in databases that satisfy specified conditions, with the template as:

```
SELECT * FROM {table} WHERE {"satisfies natural language specified condition"} LIMIT B
```

While we limit the output size to be `B = 256` to keep the total cost within a reasonable budget. The challenging scenarios are when the number of rows that satisfy the predicate is few (*i.e.*, "rare event finding"). Table 2 shows similar sets of comparison, but the metric is F1 score which evaluates the quality of `SELECT`-ed rows. Overall UQE (with claude-3-haiku as backbone LLM) consistently achieves comparable or better performance than the baseline methods. MIPS which uses the same embedding of unstructured data as UQE, has high variance across different types of queries. The queries such as "dialogs with account access issues" would be very suitable for MIPS as the embedding similarity is able to capture that well. For queries involving reasoning (*e.g.*, find the images with less than 4 objects), it is pretty hard for pretrained embeddings to express this.

6.1.3 Abstraction and aggregation

This task abstracts the intrinsics of each row, and then performs semantics-based `GROUP BY`, grouping the common intrinsics across all rows. Finally, it provides aggregated statistics over each group:

```
SELECT derived_attribute, COUNT(*) FROM {table}
GROUP BY {"extract an abstract intrinsic attribute specified in natural language"}
AS derived_attribute LIMIT 10
```

The challenging problems in this task are (i) building a taxonomy with good coverage, and (2) bias and variance reduction for groups with small population. The result of this query is a list of

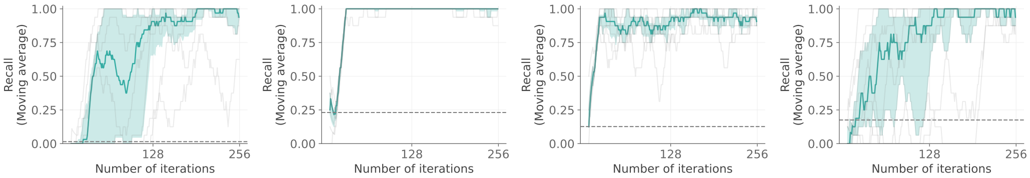


Figure 5: Recall (moving average with window size 16) against the number of iterations on (from left to right) AirDialog with condition {`cancel`, `no_flight`} and Clevr with {`obj_count < 4`, `#spheres > 3`}. Colored lines and shades denote median and interquartile ranges across 8 independent queries and gray lines denote individual queries. The gray dashed lines denote the fraction of the positive population in the entire dataset.

Table 4: Quality at different compute budget B.

	Budget B	256	128	64	32
Retrieval	latency(s)	38.08	21.61	11.11	5.84
	F1 score	0.978	0.974	0.921	0.828
Aggregation	latency(s)	-	5.83	4.28	2.93
	Error	-	5.75%	6.84%	8.29%

Table 5: Error of the aggregation operation on IMDB dataset under different budget B .

Budget B	512	256	128	64	32
UDF	8.11%	8.35%	13.67%	-	-
UQE	-	-	5.75%	6.84%	8.29%

tuples of derived attributes and their number of occurrences in the dataset. We use the earth mover's distance (EMD [30]) as the evaluation metric to compare the extracted tuples and ground-truth tuples. The distance between a pair of attributes is defined by one minus the cosine similarity of their text embeddings. We can see from Table 3 that UQE consistently outperforms baselines while achieving much lower cost. We also show in Appendix C with more qualitative results comparisons.

6.2 Ablation studies

We study the effectiveness of UQE for aggregation queries in Section 6.2.1 and non-aggregation queries in Section 6.2.2, and the quality/cost trade-off of UQE in Section 6.2.3. In appendix we provide more results on other modalities C.3, consistency C.4 and latency C.5.

6.2.1 Variance of different sampling approaches for aggregation queries

To decouple the variance introduced by the algorithm and the bias introduced by the LLM based predictors, here we use the ground-truth label as the predictive result and focus on the effectiveness of variance reduction. Figure 4 shows the box plot of different sampling methods. We can see using stratified sampling over the embeddings of unstructured content achieves significant lower variance compared to the uniform random sampling. Also both of these achieve similar expected values, which also justifies the correctness or unbiasedness.

6.2.2 Efficiency of online learning for non-aggregation queries

We show the effectiveness of the online learning in terms of the recall as a function of the iteration steps in Figure 5. Compared to the dashed line in the figure which indicates the results of uniform random sampling, the online learning can achieve significant boost in terms of the recall. While for some queries the variance at early iterations can be high, these all converge well in the end.

6.2.3 Trade-off between cost/latency and accuracy

Generally the larger compute budget B the better quality UQE will get, and we verify this in Table 4 5. UQE can achieve pretty good quality even with very low budget, and notably compared to the baseline, it achieves similar quality with 16x reduction of the compute needed. We show more results in Section C.5 regarding the compute efficiency.

7 Conclusion

This paper proposed an unstructured query engine that leverages 1) the flexibility of LLMs for data understanding; 2) the advances in sampling and online learning for efficient data scanning; 3) and the compiler that bridges these algorithmic workflows with LLMs. We demonstrated its efficiency and accuracy over three analytic tasks on four datasets with two different modalities. However the current work is still very limited in terms of 1) the semantics it lacks, including table join and other types of aggregations; 2) an automated selection of LLMs and sampling configurations; 3) and scaling to even larger databases. We hope to investigate these further in future works.

Acknowledgments and Disclosure of Funding

We thank Carsten Binnig, Howie Xu, Ras Bodik, Xinyi Chen, and the anonymous reviewers for providing helpful discussion and suggestions.

References

[1] Voyage 2. Voyage 2. https://github.com/voyage-ai/voyageai-python.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[3] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 39–48, 2015. URL https://api.semanticscholar.org/CorpusID:5276660.

[4] AI Anthropic. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 2024.

[5] Sören Becker, Johanna Vielhaben, Marcel Ackermann, Klaus-Robert Müller, Sebastian Lapuschkin, and Wojciech Samek. Audiomnist: Exploring explainable artificial intelligence for audio analysis on a simple benchmark. *Journal of the Franklin Institute*, 2023. ISSN 0016-0032. doi: https://doi.org/10.1016/j.jfranklin.2023.11.038. URL https://www.sciencedirect.com/science/article/pii/S0016003223007536.

[6] Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Prompting is programming: A query language for large language models. *Proceedings of the ACM on Programming Languages*, 7 (PLDI):1946–1969, 2023.

[7] Don Chamberlin. *A complete guide to DB2 universal database*. Morgan Kaufmann, 1998.

[8] Derek Chen, Howard Chen, Yi Yang, Alex Lin, and Zhou Yu. Action-based conversations dataset: A corpus for building more in-depth task-oriented dialogue systems. *arXiv preprint arXiv:2104.00783*, 2021.

[9] Wenhu Chen. Large language models are few (1)-shot table reasoners. *arXiv preprint arXiv:2210.06710*, 2022.

[10] Xinyun Chen, Chen Liang, Adams Wei Yu, Denny Zhou, Dawn Xiaodong Song, and Quoc V. Le. Neural symbolic reader: Scalable integration of distributed and symbolic representations for reading comprehension. In *International Conference on Learning Representations*, 2020. URL https://api.semanticscholar.org/CorpusID:212814759.

[11] Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, et al. Binding language models in symbolic languages. *arXiv preprint arXiv:2210.02875*, 2022.

[12] Edgar F Codd. *The relational model for database management: version 2*. Addison-Wesley Longman Publishing Co., Inc., 1990.

[13] Nada Elgendy and Ahmed Elragal. Big data analytics: a literature review paper. In *Advances in Data Mining. Applications and Theoretical Aspects: 14th Industrial Conference, ICDM 2014, St. Petersburg, Russia, July 16-20, 2014. Proceedings 14*, pages 214–227. Springer, 2014.

[14] Ramez Elmasri and Shamkant B Navathe. *Database systems: models, languages, design, and application programming*. Pearson, 2013.

[15] Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun Qi, Scott Nickleach, Diego Socolinsky, Srinivasan Sengamedu, and Christos Faloutsos. Large language models on tabular data–a survey. *arXiv preprint arXiv:2402.17944*, 2024.

[16] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. Text-to-sql empowered by large language models: A benchmark evaluation. *ArXiv*, abs/2308.15363, 2023. URL https://api.semanticscholar.org/CorpusID:261276437.

[17] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR, 2023.

[18] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[19] Roman Garnett. *Bayesian optimization*. Cambridge University Press, 2023.

[20] Otis Gospodnetic, Erik Hatcher, and Michael McCandless. *Lucene in action*. Simon and Schuster, 2010.

[21] Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and D. Zhang. Towards complex text-to-sql in cross-domain database with intermediate representation. *ArXiv*, abs/1905.08205, 2019. URL https://api.semanticscholar.org/CorpusID:159041042.

[22] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5*, pages 507–523. Springer, 2011.

[23] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017.

[24] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[25] Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. Table-gpt: Table-tuned gpt for diverse table tasks. *arXiv preprint arXiv:2310.09263*, 2023.

[26] Shu Liu, Asim Biswal, Audrey Cheng, Xiangxi Mo, Shiyi Cao, Joseph E Gonzalez, Ion Stoica, and Matei Zaharia. Optimizing llm queries in relational workloads. *arXiv preprint arXiv:2403.05821*, 2024.

[27] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

[28] Barzan Mozafari and Ning Niu. A handbook for building an approximate query engine. *IEEE Data Eng. Bull.*, 38(3):3–29, 2015.

[29] Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. Neural programmer: Inducing latent programs with gradient descent. *CoRR*, abs/1511.04834, 2015. URL https://api.semanticscholar.org/CorpusID:6715185.

[30] Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *2009 IEEE 12th international conference on computer vision*, pages 460–467. IEEE, 2009.

[31] Mohammad Reza Pourreza and Davood Rafiei. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *ArXiv*, abs/2304.11015, 2023. URL https://api.semanticscholar.org/CorpusID:258291425.

[32] Raghu Ramakrishnan and Johannes Gehrke. *Database management systems*. McGraw-Hill, Inc., 2002.

[33] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jeanbaptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[34] Mohammed Saeed, Nicola De Cao, and Paolo Papotti. Querying large language models with sql. *arXiv preprint arXiv:2304.00472*, 2023.

[35] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. *ArXiv*, abs/2109.05093, 2021. URL https://api.semanticscholar.org/CorpusID:237491759.

[36] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104 (1):148–175, 2015.

[37] Ruoxi Sun, Sercan O Arik, Hootan Nakhost, Hanjun Dai, Rajarishi Sinha, Pengcheng Yin, and Tomas Pfister. Sql-palm: Improved large language modeladaptation for text-to-sql. *arXiv preprint arXiv:2306.00739*, 2023.

[38] Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11888–11898, 2023.

[39] James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Halevy. Neural databases. *arXiv preprint arXiv:2010.06973*, 2020.

[40] Vertex. Vertex API. https://cloud.google.com/vertex-ai/generative-ai/docs/embeddings/get-multimodal-embeddings.

[41] Wei Wei, Quoc Le, Andrew Dai, and Jia Li. Airdialogue: An environment for goal-oriented dialogue research. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3844–3854, 2018.

[42] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1192–1200, 2020.

[43] Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. Neural enquirer: Learning to query tables. *ArXiv*, abs/1512.00965, 2015. URL https://api.semanticscholar.org/CorpusID:6715526.

[44] Pengcheng Yin, Graham Neubig, Wen tau Yih, and Sebastian Riedel. Tabert: Pretraining for joint understanding of textual and tabular data. *ArXiv*, abs/2005.08314, 2020. URL https://api.semanticscholar.org/CorpusID:218674345.

[45] Lijun Yu, Jin Miao, Xiaoyu Sun, Jiayi Chen, Alexander G Hauptmann, Hanjun Dai, and Wei Wei. Documentnet: Bridging the data gap in document pre-training. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 707–722, 2023.

[46] Tao Yu, Michihiro Yasunaga, Kai-Chou Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir R. Radev. Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. *ArXiv*, abs/1810.05237, 2018. URL https://api.semanticscholar.org/CorpusID:52979524.

[47] Tao Yu, Rui Zhang, Kai-Chou Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Z Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *ArXiv*, abs/1809.08887, 2018. URL https://api.semanticscholar.org/CorpusID:52815560.

[48] Chunhui Yuan and Haitao Yang. Research on k-value selection method of k-means clustering algorithm. *J*, 2(2):226–235, 2019.

A Proof

Proposition: *The optimal proposal distribution* p *that minimizes the variance of estimation in Eq 2 is* $p_i \propto f(\mathcal{T}_i, \texttt{cond})$*. The variance gets 0 with this proposal.*

Let's simplify the notation a bit and use $f(x)$ for $f(\mathcal{T}_i, \texttt{cond})$ and prove the variance reduction in general cases for binary function f. For the simplicity let's omit the constant $|\mathcal{T}|$ and focus on the estimation of the expectation term. If we come up with a new proposal distribution $p : \mathcal{T} \mapsto [0, 1]$ where $\sum_{x\in\mathcal{T}} p(x) = 1$, then we get a new estimator in the following form:

$$\mathbb{E}_{x\sim q}[f(x)] = \sum_{x\in\mathcal{T}} [p(x)\frac{q(x)}{p(x)}f(x)] = \mathbb{E}_{x\sim p}[\frac{q(x)}{p(x)}f(x)] \tag{4}$$

We hope this new estimator would have lower variance. Let's define $u(x) = \frac{q(x)}{p(x)}f(x)$ for the ease of notation, and look at its variance first:

$$Var_p(u(x)) = \mathbb{E}_p[u^2(x)] - \mathbb{E}_p^2[u(x)] \tag{5}$$

Since our estimator is unbiased, $\mathbb{E}_p^2[u(x)] = E_q^2[f(x)]$ and thus has nothing to do with p, let's focus on minimizing the first term. More specifically we have the optimization as:

$$\begin{aligned} \min_p \quad & \mathbb{E}_p[u^2(x)] \\ s.t. \quad & p(x) \geqslant 0, \forall x, \\ & \sum_{x\in\mathcal{T}} p(x) = 1 \end{aligned} \tag{6}$$

Let:

$$L(p, \{\lambda_x\}, \lambda_2) = \sum_{x\in\mathcal{T}} \frac{(q(x)f(x))^2}{p(x)} - \sum_{x\in\mathcal{T}} \lambda_x p(x) + \lambda_2(\sum_{x\in\mathcal{T}} p(x) - 1) \tag{7}$$

and we can find the saddle point of $\min_p \max_{\lambda_x \lambda_2}(L(p, \{\lambda_x\}, \lambda_2))$ using K.K.T condition.

$$\begin{cases} -\frac{(q(x)f(x))^2}{p(x)^2} - \lambda_x + \lambda_2 = 0, \forall x \in \mathcal{T} \\ \lambda_x p(x) = 0, p(x) \geqslant (0), \forall x \\ \lambda_2 \sum_x (p(x) - 1) = 0, \sum_x p(x) = 1 \end{cases}$$

and we can get the optimal solution of

$$p(x) = \frac{q(x)f(x)}{E_q[f(x)]} \tag{8}$$

and put it back into Eq(1) we can see the optimal variance would be

$$\begin{aligned} Var_p(u(x)) =& \sum_x \frac{(q(x)f(x))^2}{\frac{q(x)f(x)}{\mathbb{E}_q[f(x)]}} - \mathbb{E}_p^2[u(x)] \\ =& \mathbb{E}_q[f(x)] \sum_x q(x)f(x) - \mathbb{E}_q^2[f(x)] = 0 \end{aligned} \tag{9}$$

which means one sample would be good enough! In our context, $q(x)$ is usually just a constant (e.g., $q(x) = \frac{1}{\mathcal{T}}$ for the [Case Count]), and $f(x) \in \{0, 1\}$. In this case, a simplified optimal proposal would be:

$$p(x) \propto f(x) \text{ and the partition function is } \mathbb{E}_q[f(x)] \tag{10}$$

or in another word, ideally we should have zero-chance to sample from regions where $f(x) = 0$, and have equal chances to sample where $f(x) = 1$.

B UQL specifications

B.1 Tokenizer

We use the following pattern matching to tokenize the UQL programs/queries.

```
import ply.lex as lex
reserved = {
   'select' : 'SELECT',
   'from' : 'FROM',
   'where': 'WHERE',
   'as' : 'AS',
   'limit': 'LIMIT',
   'group': 'GROUP',
   'order': 'ORDER',
   'by': 'BY',
   'to': 'TO',
   'and': 'AND',
   'or': 'OR',
   'count': 'COUNT',
   'avg': 'AVG',
   'sum': 'SUM',
   'desc': 'DESC',
}
tokens = [
  'SEPARATOR',
  'ALL',
  'NL_LITERAL',
  'VAR_NAME',
  'TABLE_URL',
  'COMPARE_OPERATOR',
  'INTEGER',
  'FLOAT',
  'LEFT_PARENTHESIS',
  'RIGHT_PARENTHESIS',
] + list(reserved.values())
t_SEPARATOR = r','
t_ALL = r'\*'
t_NL_LITERAL = r'"((?:\\.|[^"\\])*)"'
t_COMPARE_OPERATOR = r'(<>|>=|<=|!=|>|<|=)'
t_INTEGER = r'[-]?\d+'
t_FLOAT = r'[+-]?[0-9]*\.[0-9]+'
t_LEFT_PARENTHESIS = r'\('
t_RIGHT_PARENTHESIS = r'\)'
def t_VAR_NAME(t):
  r'[a-zA-Z_][a-zA-Z_0-9]*(\.[a-zA-Z_][a-zA-Z_0-9]*)*'
  t.type = reserved.get(t.value.lower(), 'VAR_NAME')
  return t
```

B.2 Grammar

Below we show the context free grammar of the UQL. Note that this represents a subset of the "natural query" analogy to SQL, and we leave other clauses like table join into the future work.

```
==========================
|                        |
|       UQL grammar      |
|                        |
==========================

uql_query : select_clause from_clause
          | select_clause from_clause optional_clause_combo
```

```
optional_clause_combo : optional_clause_combo optional_clause
                      | optional_clause

optional_clause : limit_clause
                | to_clause
                | where_clause
                | group_by_clause
                | order_by_clause

select_clause : SELECT select_expression

select_expression : select_expression SEPARATOR select_literal
                  | select_literal

select_literal : ALL
               | variable_literal
               | nl_literal
               | aggregation
               | INTEGER

aggregation : agg_op LEFT_PARENTHESIS VAR_NAME RIGHT_PARENTHESIS
            | agg_op LEFT_PARENTHESIS ALL RIGHT_PARENTHESIS
            | agg_op LEFT_PARENTHESIS VAR_NAME RIGHT_PARENTHESIS AS VAR_NAME
            | agg_op LEFT_PARENTHESIS ALL RIGHT_PARENTHESIS AS VAR_NAME

agg_op : AVG
       | COUNT
       | SUM

variable_literal : VAR_NAME
                 | VAR_NAME AS VAR_NAME

nl_literal : NL_LITERAL
           | NL_LITERAL AS VAR_NAME

from_clause : FROM VAR_NAME

where_clause : WHERE where_expression

where_expression : where_expression AND predicate
                 | where_expression OR predicate
                 | predicate

group_by_clause : GROUP BY group_by_expression

group_by_expression : group_by_expression SEPARATOR group_by_literal
                    | group_by_literal

group_by_literal : variable_literal
                 | nl_literal

order_by_clause : ORDER BY order_by_expression
                | ORDER BY order_by_expression DESC

order_by_expression : order_by_expression SEPARATOR order_by_literal
                    | order_by_literal

order_by_literal : VAR_NAME
                 | NL_LITERAL
                 | INTEGER

predicate : NL_LITERAL
          | VAR_NAME COMPARE_OPERATOR NL_LITERAL
          | VAR_NAME COMPARE_OPERATOR INTEGER
```

Dataset	Conditions	percentage of occurrence in the data
Airdialog	book	51.40%
	cancel	1.46%
	no_flight	23.08%
	no_reservation	23.89%
ABCD	single_item_query	10.41%
	account_access	10.44%
IMDB	positive_review	50%
Clevr	`obj_count < 4`	12.65%
	`# spheres > 3`	17.53%

Table 6: Dataset statistics

```
            | VAR_NAME COMPARE_OPERATOR FLOAT

limit_clause : LIMIT INTEGER

to_clause : TO VAR_NAME
```

C Experiments

C.1 Datasets

We evaluate different approaches on common analytical tasks in three widely used application domains. We use the datasets that were previously created for discriminative tasks, as these datasets contain both the unstructured columns and the structured ones (the labels in the corresponding dataset). We then hiden these structured label columns and perform analytical tasks on the unstructured ones, where these hidden structured columns will be used to compute the groundtruth.

- **User review mining**. We use IMDB [27] for semantic analysis of user sentiment. The entire dataset contains 50K highly polar movie reviews with positive and negative sentiment labels.
- **Goal-oriented (customer service) dialogue systems**. We use two datasets for this category, including 1) Action-Based Conversations Dataset (ABCD [8]) for intent identification, which has 10,042 dialogs with 10 distinct user intents requiring unique sequences of actions constrained by policies to achieve task success; and 2) AirDialog [41] for conversation outcome understanding. It contains 402,037 goal-oriented conversation on flight booking with 5 possible ground-truth states.
- **Image understanding**. We use Clevr [23] dataset for multimodal data understanding and retrieving. Specifically we use the val split of the dataset, which contains 15,000 images of objects containing different number of cylinders, cubes and spheres with different sizes/colors. We down-sample the image to size no more than 128 × 128, and only feed the images to the LLMs while holding out scene metadata for evaluation only.

Also see Table 6 for the detailed statistics of different query conditions. Rare events like `cancel` in AirDialog is typically challenging to find and aggregate on.

C.2 Parameter and experiment setup

For the embeddings, we use the voyage-2 for text and for images, we use Google Vertex API with dimensionality of 512. We preprocess the embeddings for all the datasets and keep them static during the queries.

For the aggregation queries, we use faiss [2] to cluster the embeddings into 10 groups, and perform stratified sampling on top.

For the online learning setting for non-aggregation queries, in our experiment we simply set $\epsilon_{t,i}$ to be 0. We start training the function g when we collect at least one possible and one negative example labeled by the LLM. Then after every minibatch of samples collected, we train g via linear logistic regression and simply leverage sklearn for that.

Other parameters that might matter include: the sampling budget B for aggregation queries is 128 and for non-aggregation queries it is 256. For group-by queries the UQE needs a step in building the taxonomy, where the budget we use for that is 16.

[2] https://github.com/facebookresearch/faiss

Table 7: Retrieval F1 score on AudioMnist dataset.

Query	UDF-gemini	MIPS	UQE-gemini
"Three"	0.109 ± 0.022	0.445	**0.839 ± 0.135**
"A number whose square equals to itself"	0.259 ± 0.068	0.039	**0.922 ± 0.024**
"The minimum prime number"	0.107 ± 0.044	0	**0.917 ± 0.025**

Table 8: Performance of UQE with different LLM backends.

Task	UDF	UQE-haiku (in the main paper)	UQE-gpt4o-mini-0718
Retrieval	0.505 ± 0.030	0.978 ± 0.003	0.956 ± 0.010
Aggregation	13.67% ± 6.24%	5.75% ± 3.43%	6.33% ± 4.71%

We set these parameters based on educated guess and keep them as default across all the queries over all the datasets.

C.3 Audio modality

We use the audio MNIST [5] data for this experiment. This dataset contains 30k wav files from 60 speakers pronouncing digits 0-9. We perform the audio semantic retrieval experiments. The query is first converted to audio space using TTS and the corresponding audio embedding is used for MIPS search. We use Gemini Pro 1.5 as the backend model, as it supports the audio inputs nicely. As is shown in Table 7, the proposed UQE consistently does better than alternatives, and it also allows complex queries that require reasoning, while embedding based MIPS is limited to certain types of queries.

C.4 Consistency of the execution results

While UQE is able to leverage the foundation models to analyze the unstructured databases directly, the execution result is not deterministic due to the nature of the stochasticity of the algorithm and the LLMs themselves. In our paper we aim at reducing the variance so as to improve the consistency. However when the backend LLM gets updated, it might also cause the potential inconsistency. Below we analyze the effect of different LLM backends.

Table 8 shows the effect on the IMDB dataset, where we see little variation. Actually the difference caused by the model switching is much lower than using a worse query engine. Of course, this behavior shift would be task related, but with the advances of LLMs we believe this variation would converge and be more stable to different prompts in the future.

C.5 Latency

We report the runtime of UQE with claude-3-haiku as backbone, and lc-gpt-4-turbo as the baseline method in Table 9. We can see UQE achieves low latency in aggregation operations, but higher latency in retrieval. This is due to the online update and re-evaluation of the g function described in Section 4.1.2. The experiments were run on MacBook Pro CPU, so we expect this bottleneck would be alleviated with better engineered system, which we will focus in our future works.

C.6 Group By qualitative results

For `single_item_query` in ABCD, the items mentioned in the dialogs found by UQE is:

```
[['boots' '338']
 ['jacket' '282']
 ['jeans' '268']
 ['shirt' '190']]
```

For `account_access` in ABCD, the issues mentioned in the dialogs found by UQE is:

```
[['Forgot Password' '457']
 ['Forgot Username' '406']
 ['Lost phone for two-factor authentication' '362']]
```

Which is very close to the ground truth (recover_username, reset_2fa, recover_password).

For airdialog, the outcomes found by UQE is

Table 9: Runtime (in seconds) comparison for different types of queries over different benchmarks.

Methods	Conditional aggregation				Semantic Retrieval			
	Clevr	ABCD	IMDB	Airdialog	Clevr	ABCD	IMDB	Airdialog
UQE-claude-3-haiku	3.13	3.34	5.83	3.85	46.00	41.20	38.08	67.14
lc-gpt-4-turbo	28.06	4.72	4.37	3.23	63.38	10.10	20.61	23.06

```
[['Flight Ticket Booked' '199383']
 ['No Flights Available' '100884']
 ['No Reservation Found' '80775']
 ['Flight Reservation Cancelled' '51937']]
```

Where the ground truth has one more additional outcome (cancel). But since the percentage of cancellation is very small, it is expected that this might be missing from the group by abstraction when number of occurs are very limited.

D Prompts

D.1 Prompts for lc-LLMs

D.1.1 Task: Conditional aggregation

IMDB dataset

System prompt

```
Read the following movie reviews, and categorize them into either positive
or negative class, depending on the sentiment of the review.If the movie
has a mixed sentiment, try your best to classify into positive or negative
class based on the overall sentiment.In the end, please just output a
single number, which is [the total number of positive reviews]
```

User prompt

```
Below are the reviews:
[Review 0]: Some TV programs continue into embarrassment (my beloved 'X-
    ↪ Files' comes to mind.)...
[Review 1]: The tale of the titular Adam (Mark O' Halloran) and Paul (Tom
    ↪ Murphy), ...
```

ABCD dataset

System prompt

```
The following dialogs between a customer service agent and a customer.
Dialogs start by headers such as **Dialog 1**, **Dialog 2**, and so on.
Your task is to classify whether the dialog content is about the theme
"account access issue". Then count how many dialogs are talking about this
theme. In the end, output the count as a single number.
Here is more detailed explanation about Theme "<THEME>". Be sure to use
this information when you classify.
Theme "<THEME>" dialogs content is about THEME_EXPLANATION. Please perform
thorough analysis for each of the dialog. In the end, please **only**
output a single number, which is [the total number of dialogs] that talks
about the theme "account access issue".
```

```
<THEME> = account access issue
<THEME> = requesting detailed specifications of a certain item sold on the
website
```

```
<THEME_EXPLANATION> = the customer could not access the account and was
locked out, such as couldnt́ recall their username, couldn't perform
two-factor authentication, or forgot their password and couldn't access
their account
```

```
<THEME_EXPLANATION> = the customer needed help with the detailed
specifications of a certain item sold on the website, such as inquiries
about detailed information of a specific retail item about materials,
whether it shrinks, stock availability, etc., but NOT inquiries about
promotions, order status, shipping status, questions about how to use the
website to purchase an item, or difficulties on using the website to add to
cart or purchase an item, or inquiries about subscription
```

User prompt

```
Below are the dialogs:
**Dialog 0**:
[agent]: Hello, how can i help you today
[customer]: Hello my name is Alessandro Phoenix and I need to make sure
the shipping cost is included on my order
...
```

AirDialog dataset

System prompt

```
The following are dialogs between a airline ticketing agent and a customer.
Dialogs start by headers such as **Dialog 1**, **Dialog 2**, and so on.
The outcome of the dialog will be one of the following 5 categories:
[book]: the agent has booked a flight for the customer (not including the
flight change);
[cancel]: the agent canceled the existing valid reservation for the
customer;
[no_reservation]: the customer wants to change or cancel the flight 1037
but there is no valid reservation under this customer;
[no_flight]: the customer aims to book a flight from departure to
destination but finds no flights between departure and destination;
Your task is to count how many dialogs have outcome <OUTCOME>. Please
**only** output a single number, which is [the total number of dialogs]
that satisfied the above requirements.
```

```
<OUTCOME> = [book]
<OUTCOME> = [cancel]
<OUTCOME> = [no_reservation]
<OUTCOME> = [no_flight]
```

User prompt

```
Below are the dialogs:
**Dialog 0**:
customer: Hello.
agent: Hello, how can I help you today?
customer: Can you please find a flight from DFW to SEA?
...
**Dialog 1**:
customer: Hi, I am Melissa Thompson.
agent: Hello, how may I support you today?
customer: I want to celebrate Thanks giving day 11/23 with my friends
at New York. Can you book a ticket for me?
...
```

Clevr dataset

Task: Conditional aggregation

System prompt

```
Please read the following images, and count how many of them show that
<CONDITION>.
```

```
<CONDITION> = there are less than 4 objects in the image
<CONDITION> = there are more than 3 spheres in the image
```

User prompt

```
image\_0: <base64\_encoded\_image>
image\_1: <base64\_encoded\_image>
...
Please output a single number, which is the total number of images that
    ↪ satisfy the condition.
```

D.1.2 Task: Semantic retrieval

IMDB dataset

System prompt

```
Read the following movie reviews, and list the indices of reviews with
positive sentiment.  If the movie has a mixed sentiment, try your best to
classify into positive or negative class based on the overall sentiment.In
the end, please only output a list of indices in the format of [review_3,
review_7, ...]
```

User prompt

```
Below are the reviews:
[Review 0]: Some TV programs continue into embarrassment (my beloved 'X-
    ↪ Files' comes to mind.)...
[Review 1]: The tale of the titular Adam (Mark O' Halloran) and Paul (Tom
    ↪ Murphy), ...
```

ABCD dataset

System prompt

```
The following are dialogs between a customer service agent and a customer.
Dialogs start by headers such as **dialog_1**, **dialog_2**, and so on.
Your task is to find out the dialogs where <CONDITION>.  Please only output
a list of indices in the format of [dialog_3, dialog_7, ...]
```

```
<CONDITION> = the customer could not access the account and was locked
out, such as couldnt́ recall their username, couldn’t perform two-factor
authentication, or forgot their password and couldn’t access their account
<CONDITION> = the customer needed help with the detailed specifications
of a certain item sold on the website, such as inquiries about detailed
information of a specific retail item about materials, whether it shrinks,
stock availability, etc., but NOT inquiries about promotions, order status,
shipping status, questions about how to use the website to purchase an item,
or difficulties on using the website to add to cart or purchase an item, or
inquiries about subscription
```

User prompt

```
Below are the dialogs:
**dialog_0**:
[agent]: Hello! Welcome to AcmeBrands, how cani help you?
[customer]: Hi. I am very frustrated because I am trying to use your
    ↪ website and it is running SO slowly!
...
**dialog_1**:
[customer]: hi there
[agent]: Hi! What can I help you with today?
[customer]: i wanted to know if you'd be able to tell me the arm length on
    ↪ a shirt i'm thinking of buying?
...
```

AirDialog dataset

System prompt

```
The following are dialogs between a airline ticketing agent and
a customer.  Dialogs start with headers such as **dialog_1**, **dialog_2**,
and so on.  Your task is to find out the dialogs where CONDITION. Please
only output a list of indices in the format of [dialog_3, dialog_7, ...]
```

```
<CONDITION> = the agent has booked a flight for the customer (not including
the flight change)
<CONDITION> = the agent canceled the existing valid reservation for the
customer
<CONDITION> = the customer aims to book a flight from departure to
destination but finds no flights between departure and destination
<CONDITION> = the customer wants to change or cancel the flight but there
is no valid reservation under this customer
```

User prompt

```
Below are the dialogs:
**Dialog 0**:
customer: Hello.
agent: Hello, how can I help you today?
customer: Can you please find a flight from DFW to SEA?
...
**Dialog 1**:
customer: Hi, I am Melissa Thompson.
agent: Hello, how may I support you today?
customer: I want to celebrate Thanks giving day 11/23 with my friends
at New York. Can you book a ticket for me?
...
```

Clevr dataset

System prompt

```
Please read and parse the following images.  Images start with labels such
as **image_0**, **image_1**, and so on.
Your task is to find out the images where <CONDITION>.  Please only output
a list of indices in the format of [image_3, image_7, ...]
```

```
<CONDITION> = there are less than 4 objects in the image
<CONDITION> = there are more than 3 spheres in the image
```

User prompt

```
image_0: <base64_encoded_image>
image_1: <base64_encoded_image>
...
Given above, the relevant images are:
```

D.1.3 Task: Abstraction and aggregation

ABCD dataset

System prompt

```
The following are dialogs between a customer service agent and a customer.
Dialogs start with headers such as **dialog_1**, **dialog_2**, and so on.
Your task is to analyze all the dialogs, and summarize
"<ABSTRACT_ATTRIBUTE>" into groups.  Please output the table of
your analysis, in the format of pairs of ("<ABSTRACT_ATTRIBUTE>",
number_of_dialogs belong to that).  Specifically in the format as:
group 1,number_of_dialogs
group 2,number_of_dialogs
...
```

```
<ABSTRACT_ATTRIBUTE> = the type of account access issue <ABSTRACT_ATTRIBUTE>
= the single item involved in the dialog
```

User prompt

```
Below are the dialogs:
**dialog_0**:
[agent]: good afternoon, how can I help you?
[customer]: hey think i mixed up or forgot which username I'm in with you
    ↪ guys as
...
**dialog_1**:
[agent]: Hi there, thanks for contacting Acme! How can I help you?
...
```

AirDialog dataset

System prompt

```
The following are dialogs between a airline ticketing agent and a customer.
Dialogs start with headers such as **dialog_1**, **dialog_2**, and so on.
Your task is to analyze all the dialogs, and summarize
"<ABSTRACT_ATTRIBUTE>" into groups.  Please output the table of
your analysis, in the format of pairs of ("<ABSTRACT_ATTRIBUTE>",
number_of_dialogs belong to that).  Specifically in the format as:
group 1,number_of_dialogs
group 2,number_of_dialogs
...
```

```
<ABSTRACT_ATTRIBUTE> = the outcome of the dialog
```

User prompt

```
Below are the dialogs:
**dialog_0**:
customer: Hi.
agent: Hello. How may I help you?
customer: I need to book a flight ticket from DEN to EWR to enjoy music
    ↪ festivals.
...
**dialog_1**:
customer: Hi.
agent: Hello, how may I help you?
...
```

D.2 Prompts for UQE-orchestrated LLMs

D.2.1 Task: Conditional aggregation, Semantic retrieval

IMDB dataset

System prompt

```
Please analyze the following movie review, and only reply <True> if
<WHERE_CLAUSE>, or <False> otherwise.
```

```
<WHERE_CLAUSE> = the review sentiment is overall positive
```

User prompt

```
[Movie review]:  May I please have my $13.00 back?  I would have rather
watched "Hydro- Electric Power Comes to North America"...
```

ABCD dataset

System prompt

Read the following customer support dialog between an agent and a customer, and only reply <True> if <WHERE_CLAUSE>, or <False> otherwise.

<WHERE_CLAUSE> = the customer could not access the account and was locked out, such as couldnt́ recall their username, couldn't perform two-factor authentication, or forgot their password and couldn't access their account
<WHERE_CLAUSE> = the customer needed help with the detailed specifications of a certain item sold on the website, such as inquiries about detailed information of a specific retail item about materials, whether it shrinks, stock availability, etc., but NOT inquiries about promotions, order status, shipping status, questions about how to use the website to purchase an item, or difficulties on using the website to add to cart or purchase an item, or inquiries about subscription

User prompt

[Dialog]: [agent]: Hi! Thank you for contacting us today. How can I help you?
[customer]: I'm pretty upset that a jacket that I ordered is now saying that it is out of stock. Do you know when it will be back in stock?
[agent]: I am so sorry that happened to you
[agent]: Yes, let me look in to that for you
[action]: Searching the FAQ pages ...
...

AirDialog dataset

System prompt

Read the following airline ticketing dialog between the customer and the agent, and only reply <True> if <WHERE_CLAUSE>, or <False> otherwise.

<WHERE_CLAUSE> = the agent has booked a flight for the customer (not including the flight change)
<WHERE_CLAUSE> = the agent canceled the existing valid reservation for the customer
<WHERE_CLAUSE> = the customer aims to book a flight from departure to destination but finds no flights between departure and destination
<WHERE_CLAUSE> = the customer wants to change or cancel the flight but there is no valid reservation under this customer

User prompt

[Dialog]: customer: Hello.
agent: Hello, how can I help you?
customer: Please book a flight ticket from CLT to DEN.
agent: Sure, let me know your travelling dates.
...

Clevr dataset

System prompt

Read the following image, and only reply <True> if <WHERE_CLAUSE>, or <False> otherwise.

<WHERE_CLAUSE> = there are less than 4 objects in the image
<WHERE_CLAUSE> = there are more than 3 spheres in the image

User prompt

Image: <base64_encoded_image>

D.2.2 Task: Abstraction and aggregation

ABCD dataset

1. **Building taxonomy**

System prompt

```
The following are dialogs between a customer service agent and
a customer.  Dialogs start with headers such as **dialog_1**,
**dialog_2**, and so on.
Your task is to analyze all the dialogs, and summarize
"<ABSTRACT_ATTRIBUTE>" into groups.  Please output the table of
your analysis, in the format of pairs of ("<ABSTRACT_ATTRIBUTE>",
number_of_dialogs belong to that).  Specifically in the format as:
group 1,number_of_dialogs
group 2,number_of_dialogs
...
<ABSTRACT_ATTRIBUTE> = the type of account access issue
<ABSTRACT_ATTRIBUTE> = the single item involved in the dialog
```

User prompt

```
Below are the dialogs:
**dialog_0**:
[agent]: good afternoon, how can I help you?
[customer]: hey think i mixed up or forgot which username I'm in
    ↪ with you guys as
...
**dialog_1**:
[agent]: Hi there, thanks for contacting Acme! How can I help you
    ↪ ?
...
```

2. **Group-wise conditional aggregation**

System prompt

```
Read the given airline ticketing dialog between an agent and a
    ↪ customer, and classify issue, into one or several
    ↪ categories below. Here are the description of the 2
    ↪ categories:
[0]: Forgot Username
[1]: Forgot PasswordOnly reply the index of the category,
    ↪ separated by ",". Here is the example format:
[0, 3]
```

User prompt

```
Here is the customer support dialog:
[agent]: Hello, how can i help you today
[customer]: Hi. I seem to have forgotten my username
[agent]: Okay lets get that for you, could i get your Full Name
    ↪ Zip Code Email Adress and Phone Number please
[customer]: Sanya Afzal
```

AirDialog dataset

1. **Building taxonomy**

System prompt

```
The following are dialogs between a airline ticketing agent and
a customer.  Dialogs start with headers such as **dialog_1**,
**dialog_2**, and so on.
Your task is to analyze all the dialogs, and summarize
"<ABSTRACT_ATTRIBUTE>" into groups.  Please output the table of
your analysis, in the format of pairs of ("<ABSTRACT_ATTRIBUTE>",
number_of_dialogs belong to that).  Specifically in the format as:
group 1,number_of_dialogs
group 2,number_of_dialogs
...
<ABSTRACT_ATTRIBUTE> = the outcome of the dialog
```

User prompt

```
Below are the dialogs:
**dialog_0**:
customer: Hi.
agent: Hello. How may I help you?
customer: I need to book a flight ticket from DEN to EWR to enjoy
    ↪  music festivals.
...
**dialog_1**:
customer: Hi.
agent: Hello, how may I help you?
...
```

2. **Group-wise conditional aggregation**

System prompt

```
Read the given airline ticketing dialog between the customer and
    ↪ the agent, and classify outcome, into one or several
    ↪ categories below. Here are the description of the 2
    ↪ categories:
[0]: Reservation cancelled
[1]: Ticket bookedOnly reply the index of the category, separated
    ↪  by ",". Here is the example format:
[0, 3]
```

User prompt

```
Here is the airline ticketing dialog: customer: Hello
agent: Hello, how may I help you?
customer: Can you help me to book a flight ticket from SEA to AUS
    ↪ ?
agent: Sure, we are glad to help you. May I know your travelling
    ↪ dates?
```

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We provided thorough experiments and explanations of the algorithms to jusify the claim.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed that in the conclusion section and we pointed out three major limitations.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We have the proof for the optimal variance reduction proposal in the appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We use public LLMs on public benchmarks, and the prompts are all included in the appendix. For our work we have provided enough details for reimplementation, and we will work on open sourcing as well.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The data and prompts are all public and we have provided the information. We are working on open sourcing the code after going through the internal approval process.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We have provided the information in the appendix and main paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We provided the variance of the results in almost all tables when applicable.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We mainly use public API.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We have checked.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This work is a pure algorithmic and enginnering study for new forms of databases.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: No risk as far as we can see.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have properly cited the models (GPT4, Claude3) we used in the paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: No new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Not involved.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: no crowdsourcing nor research

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.